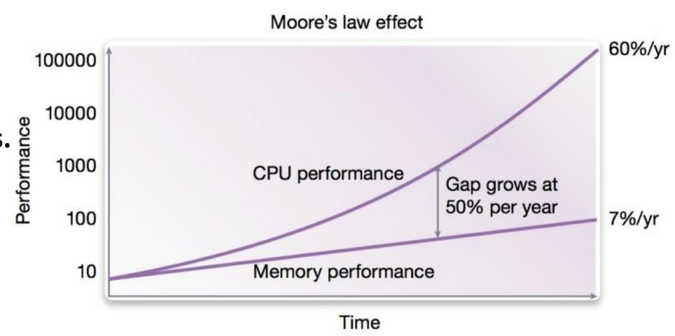
Memory First : A Performance Tuning Strategy Focusing on Memory Access Patterns

Naoki EBATA†, Ryusuke EGAWA†, Yoko ISOBE†, Ryoji TAKAKI*, and Hiroyuki TAKIZAWA†

†Tohoku University, Email: {ebata.naoki.r7@dc. , isobe@, egawa@, takizawa@}tohoku.ac.jp
*Japan Aerospace Exploration Agency, Email: ryo@isas.jaxa.jp

Background

- ✓ "Memory wall" has become one of the most worrisome issues in HPC.
  - ➢ **Sustained memory bandwidth potentially limits the performance of real applications.**
    - ◆ Cannot timely provide enough data to fully utilize a huge amount of available computational resources.
- ✓ Memory-aware code tuning is crucial for exploiting the performance of modern HPC systems.



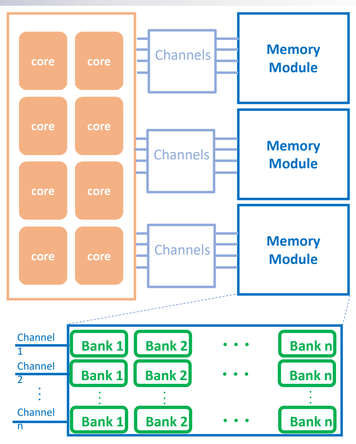
http://www.extremetech.com/wp-content/uploads/2014/07/140364245678419.jpg

Memory-Aware Code Tuning is needed

High Bandwidth Memory

- ✓ To overcome the memory wall problem, high bandwidth memory such as HMC and HBM plays key roles in modern HPC systems.
- ✓ High bandwidth memory consists of multiple channels, modules, and banks for achieving a higher memory bandwidth by the interleaved data transfers.
  - ☐ EX, NEC SX-Aurora TSUBASA (**SX-AT**) and NVIDIA Tesla V100 have 6 and 4 HBM modules, respectively.

In spite of their high peak bandwidths, high "sustained" bandwidths cannot be attained without careful code tuning.



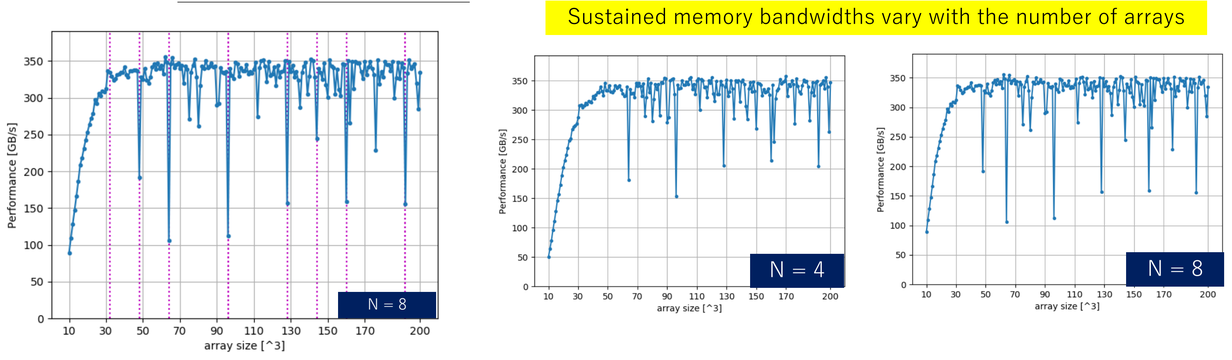
Need for Memory-Aware Code Optimization

Sustained memory bandwidth (STREAM Triad) evaluated with changing the number of arrays being accessed

- ✓ SX-AT has in total **48** channels, each of which is connected to **32** banks.

Evaluation codes

```
1 for(i=0; i<array_size; i++){
2         c_1[i] = a_1[i] + b_1[i] * scalar;
3         c_2[i] = a_2[i] + b_2[i] * scalar;
4
5         ...
6
7         c_N[i] = a_N[i] + b_N[i] * scalar;
8 }
```

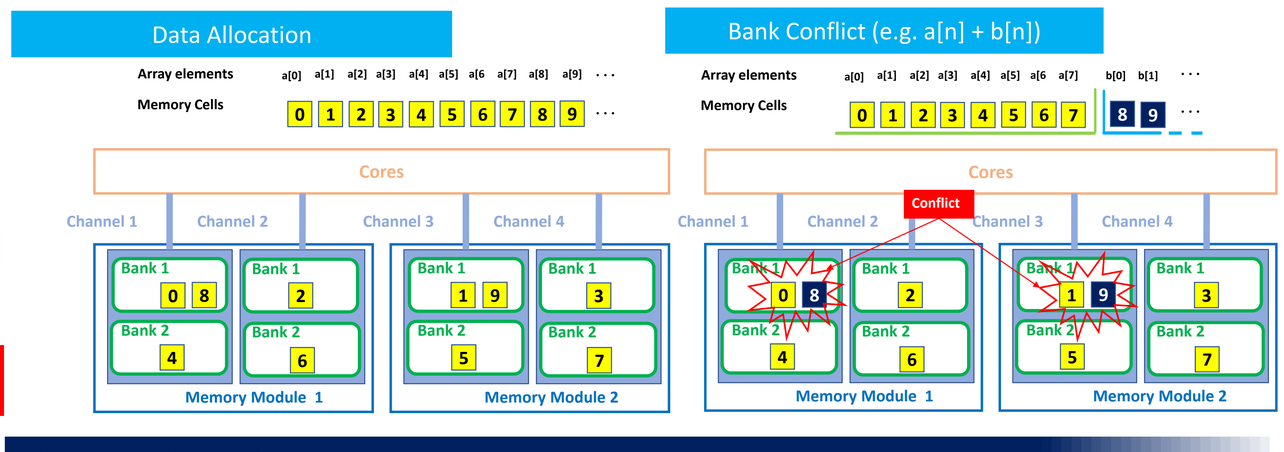


Sustained memory bandwidths vary with the number of arrays

Spike-like severe performance drops occur when the array size is a multiple value of 48 (the number of channels) or 32 (the number of banks), because memory access conflicts frequently occur.

Access Conflicts in Modern HPC Memory Subsystem

- ✓ Data allocation among memory modules and banks
  - ➢ Data (array elements) are allocated to the memory cells in a round-robin manner with the following priority.
    - ◆ Different Channels > Different Banks
- ✓ A conflict happens upon accesses to the same channel or bank .
  - ☐ Since memory accesses to the same channel or bank should be processed serially, a subsequent memory access should wait until its preceding access is completed.



Memory First !

Stable performance on memory-intensive kernels by always accessing as many memory channels and banks as possible.

Memory First = a "memory-centric" performance tuning approach

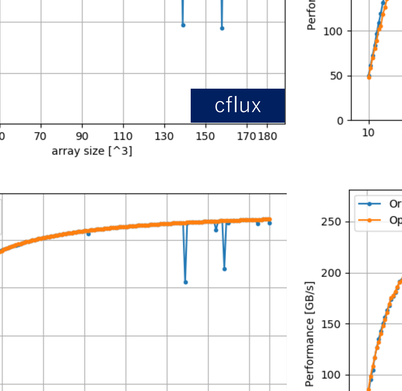
1. Write a **tiny benchmark code**, which is similar to the target kernel and capable of efficiently using the system's memory bandwidth.
   (In the CFD field, the kernel is likely either **stream** or **stencil**.)
2. Modify the tiny code so as to work as the target kernel.

→ The tiny code is already considered to make good use of memory bandwidth

= Minimizing the modification for memory awareness

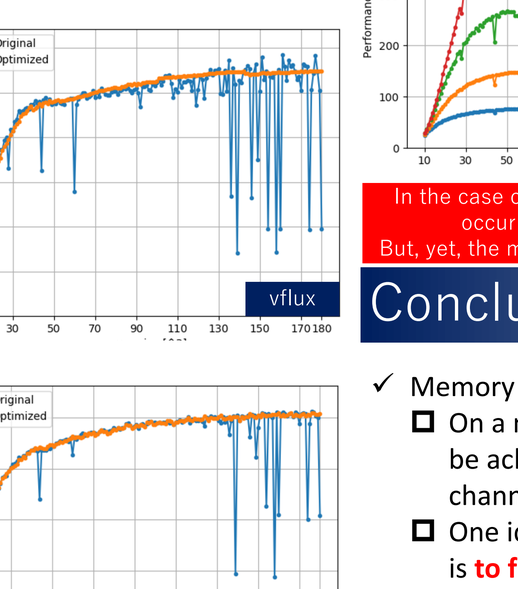
Traditional performance tuning steps
1. Write a whole application correctly computing the results
2. Optimize the code so as to exploit the system performance

→ The second step (memory-aware tuning) often needs major code modifications (e.g. data layout optimization)

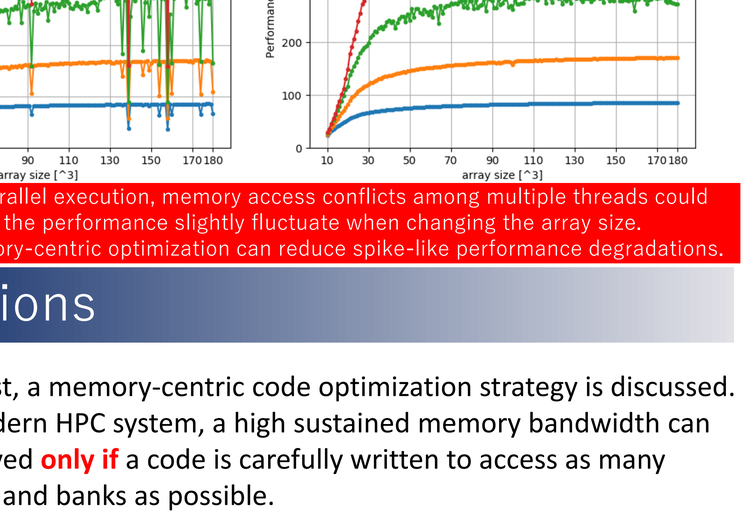


Case Study

- ✓ In the case of the Memory First strategy, a tiny code not causing conflicts on the target system is first developed, and **its data layout is always used for any array sizes**.
- ✓ In addition to basic loop optimizations for enhancing vectorization, another key to extracting the SX-AT performance is to avoid memory access conflicts (bank conflicts).



Can we use A's data layout also for B?



Data Layout optimization

Evaluation Results

Evaluation Setup

System: NEC SX-Aurora TSUBASA Type-10B

Number of cores	8
Core Performance	268 Gflop/s
Clock Frequency	1.4 GHz
Total Peak Performance	2.15 Tflop/s
Memory Capacity	48GB (6 HBM Modules)
	Each HBM module has 8 Channels, and each channel has 32 Banks
Memory Bandwidth	1228 GB/s
	48 Channels x 16 Bytes/clock x 1.6 GHz
VEOS	VEOS 1.3.2
Compiler	NEC Fortran compiler (nfort) 1.6.0

Application: UPACS-Parts

Basic Kernels of CFD code, UPACS, developed at JAXA

Kernel	Computation	Eq. term	Prog. Type	# of arrays
cflux	Flux	Convection	Stream	15
vflux	Flux	Viscous	Stream	21
muscl	Physical quantity complementation of cell surface	Convection	Stencil	15
cfacev	Differentiation of cell surface	Viscous	Stencil	21

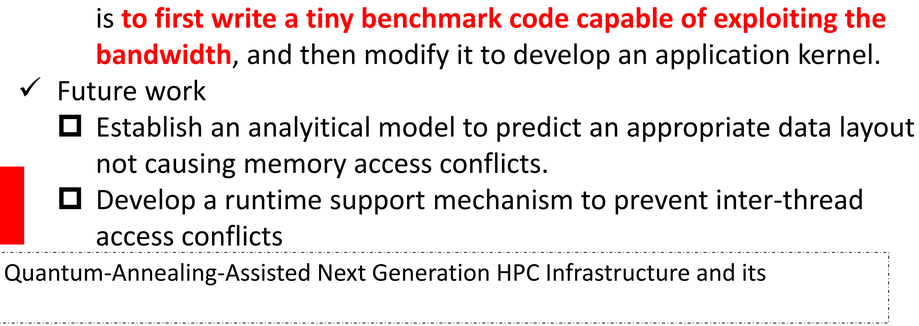
Evaluation results

The array sizes which can achieve stable performance is already known to be 181^3 by preliminary evaluations.

Single thread execution

By specifying an appropriate array size, spike-like performance degradations can successfully be removed.

Multi threads execution of cflux



In the case of parallel execution, memory access conflicts among multiple threads could occur and the performance slightly fluctuate when changing the array size.
But, yet, the memory-centric optimization can reduce spike-like performance degradations.

Conclusions

- ✓ Memory First, a memory-centric code optimization strategy is discussed.
  - ☐ On a modern HPC system, a high sustained memory bandwidth can be achieved **only if** a code is carefully written to access as many channels and banks as possible.
  - ☐ One idea to achieve high sustained performance at a low tuning cost is **to first write a tiny benchmark code capable of exploiting the bandwidth**, and then modify it to develop an application kernel.
- ✓ Future work
  - ☐ Establish an analyitical model to predict an appropriate data layout not causing memory access conflicts.
  - ☐ Develop a runtime support mechanism to prevent inter-thread access conflicts

This work is partially supported by MEXT Next Generation High-Performance Computing Infrastructures and Applications R&D Program "R&D of A Quantum-Annealing-Assisted Next Generation HPC Infrastructure and its Applications" and Grant-in-Aid for Scientific Research(B) #16H02822 and #17H01706.